Introduction to computation

Language comes in a variety of forms. We have natural languages, for instance English and Thai. Mathematics is also a language, in fact arguably the language of all branches of science. Music is a language and, as some may say, that which resemble brain waves and thus closest to our mind. Computer language is the mean by which we communicate with a computer, to make it perform the tasks we want it do.

Native to the hardware of computer is machine language. This has an alphabet comprising of only two letters, 0 and 1. Replacing this alphabet with a bigger one, that is one made up of roman characters and numbers, assembly language is analogous to machine language but is much easier for human to read. A high-level language like C uses commands and functions each one of which is in fact a macro or group of instructions in assembly language.

High-level languages can be unstructural, structural, or object-oriented depending on their main characteristics. Characteristic to unstructural languages are the discontinuous jumps, for example `goto` in Basic, or *case* and *break* in C. A noticeable trait of structural language is the existence of conditional sentences beginning with *if*, *for*, or *while*.

Both the structural and the unstructural languages are procedural, that is they are oriented around procedures. Data flow into procedures the outcome of which are outputs. Object-oriented language, on the other hand, thinks differently. As the name implies it is oriented around objects, that is the data. Functions are applied to these objects to transform them in the way the programmer wants. Because the data do not flow around their security can be better ensured by building walls around them and have all inputs to them screened. In short, an object-oriented language is one in which data have been invidualised. They thus have a character, a gene, which can be passed down from generation to generation through the method called *inheritance*.

A problem-solving machine, in other words a computing machine, was developed out of two basic requirements. Firstly, such machine should not store all solutions of all possible problems. Secondly, the speed of its information processing should be finite. Based on these requirements an abstract model of a computer was developed in 1936 by an English mathematician Alan M Turing (1912–54). It is now called the *Turing machine*. A Turing machine has three components, namely a *processor*, a *read-write head* and a *memory tape*. Comparing this with a man working out his problems on a piece of paper these represent the brain, the pencil and the piece of paper. In this conceptual model the processor works like a black box, that is to say we neglect for now its internal mechanism. In practice it is a state machine which is hard-wired.

There are four operations possible in a Turing machine. They are,

1. Write or read
2. Shift-left the head
3. Shift-right the head
4. Stop. Turing managed to prove that there exists a *universal* Turing machine capable of performing all the computation that could be done by any Turing machine.

Kurt Gödel (1906–78) demonstrated in 1931 that the Turing machine can solve a problem only if it stop. If the machine does not stop when process a certain problem, that problem is said to be *unsolvable* or *undecidable*.

Also, a Turing machine has a limit in speed but no limit in time. Its space (memory)

resource is also limitless because its tape is said to be infinitely long. In reality computers have limited resources, both space and time. They are know as *finite-state machines*. Therefore there are problems which are practically *intractable* simply because there is not enough space to contain the data or results. Not keeping solutions to all possible problems in the world is a practicality, it is also practically necessary to have no data too large to be fitted into our piece of paper.

C Compilers

Because C was developed for Unix, all subsequent compilers of the language have invariably inherited traits characteristic of the latter. Examples of non-Unix C compilers are Microsoft C and Turbo C. Among Unix related ones is GNU C Compiler (`gcc`). GNU came from the concept of Richard Stallman who was the founder of Free Software Foundation. GNU licence demands that a programmer must make source code of his programme available. This licence is sometimes called *copyleft*, in contrast with copyright. The name GNU is an interesting name. It is an acronym, and stands for GNU is Not Unix. This self-recursive idea has been found in literature as world within world, for example the play-within-play scenes in Shakespeare's *A Midsummer Night's Dream* and the book-within-book theme in Jostein Gaarder's *Sophie's Worlds* (1991).

C Programming

The word 'programme' is English while its counterpart 'program' is American, similarly 'disc' and 'disk' in the same order. But 'diskette' seems to be the same for both, and 'programming' is equally correctly derived from both 'program' and 'programme'.

Programming is an applied science, and is also very much problem-oriented. It draws resources from virtually all branches of humanly knowledge. The aim of all programmes is to solve some problem, be it how to put some words on screen or how to rescue some spacecraft some millions of mile from Earth. Examples are numerous, but some of the things one often turns to in the course of writing a computer programme are Geometry if the problem at hand is a geometrical one, Calculus if it needs numerical method approach and Mathematical Programming if it is an optimisation problem.

Like English in the physical world C, through its associates with Unix, has become the *lingua franca* of the internet and computing world. This is why we begin our first course in computer programming with the study of this language. Speakers of C are likely to find in their future career similarities to C in many other places, for instance the language used for writing the macros in Matlab.

Bibliography

John P Hayes. *Computer architecture and organization.* 2nd ed, McGraw-Hill, 1988